Data Mining Assignment 1: Classification

Yassine Akrim

April 12, 2023

1 Introduction

The objective of this assignment is to identify a subset of people who were most likely to respond positively to a promotion and generate the highest profit for the company. To do that, we are given two datasets, **'existing-customers.xlsx'** and **'potential-customers.xlsx'**, and we are tasked to clean them (dealing with missing values, normalization...) to finally predict the list of potential customers and expected revenue at the end.
This assignment has been solved using python, with the help of a variaty of libraries including 'pandas, seaborn, matplotlib' for data manipulation and vizualisation, and 'sklearn' for everything related to machine learning.

2 The project

2.1 Data Exploration and Preprocessing

First, we start by exploring the given datasets, starting by "existing-customers.xlsx" to get an idea of what type of data we are dealing with, and if there are any missing values. See Figure 1

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       30725 non-null  object
 2   education       32561 non-null  object
 3   education-num   32561 non-null  int64
 4   marital-status  32561 non-null  object
 5   occupation      30718 non-null  object
 6   relationship    32561 non-null  object
 7   race            32561 non-null  object
 8   sex             32561 non-null  object
 9   capital-gain    32561 non-null  int64
 10  capital-loss    32561 non-null  int64
 11  hours-per-week  32561 non-null  int64
 12  native-country  31978 non-null  object
 13  class           32561 non-null  object
dtypes: int64(5), object(9)
memory usage: 3.5+ MB
```

Figure 1: Info about the "existing-customers.xlsx" file.

We notice a few columns are missing values, and we can highlight that using a seaborn heatmap to get a general idea. See Figure 2. We notice 3 columns (workclass, occupation, and native-country) have a few missing values.

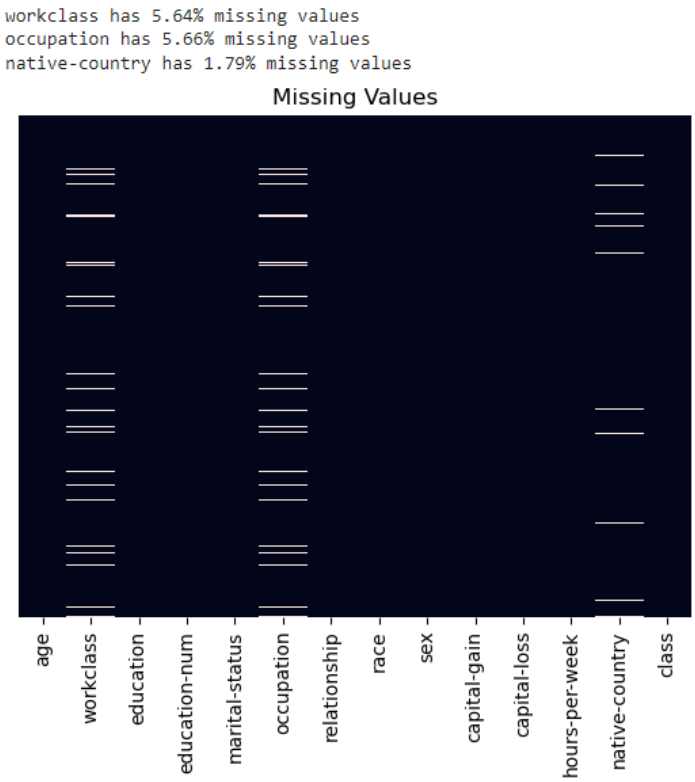
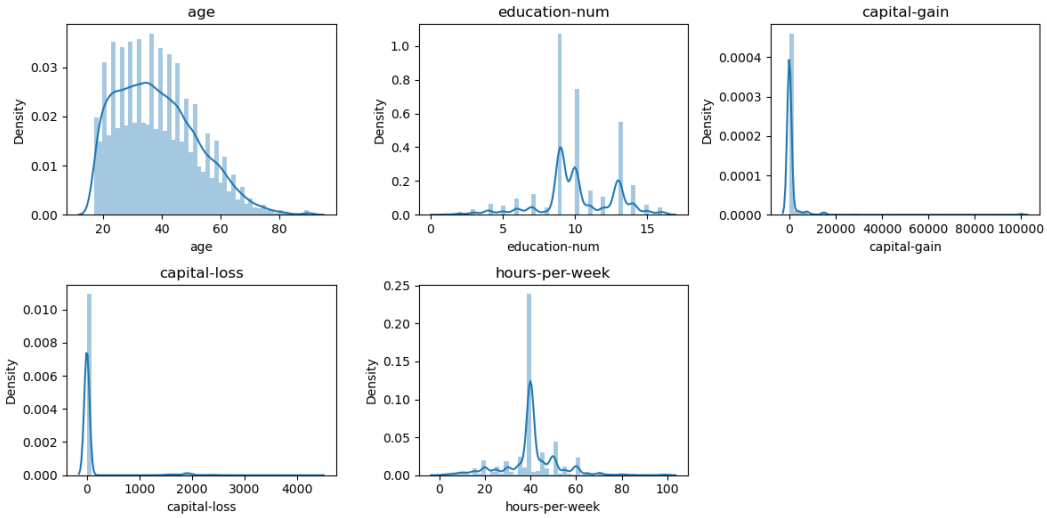


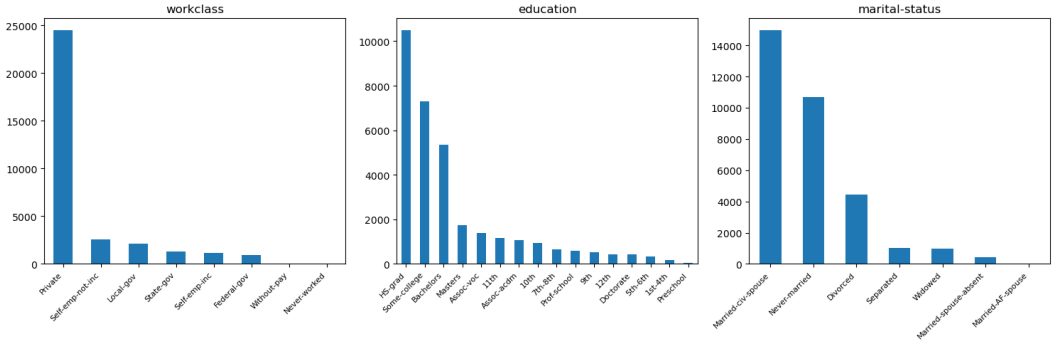
Figure 2: Vizualizing the missing data from the dataset.

There are various methods for imputing missing values, and the best approach depends on the specific data and variables. However, in this case, the columns with missing values were all categorical, which made it more challenging to use some imputation methods like the KNN imputer. To use KNN imputer or other similar approaches, we would have had to convert all the other variables into numerical ones, which would have been a complex process. Therefore, the approach we took of imputing the missing values with the most frequent value of their respective columns is a common practice in cases of categorical columns. And when looking at the results, we can see that using most frequent value performed quite well, had this approach not performed well, we would have explored other imputation options based on the specific data and variables.

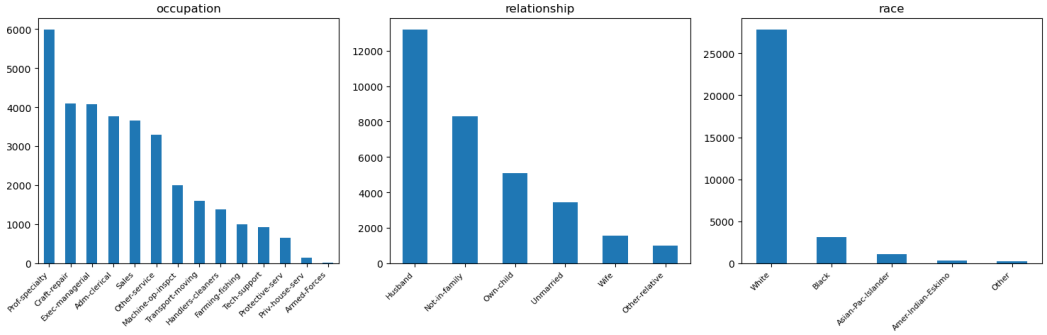
Next we look at the data distribution for both numerical and categorical columns and we get the following. See Figure 3.



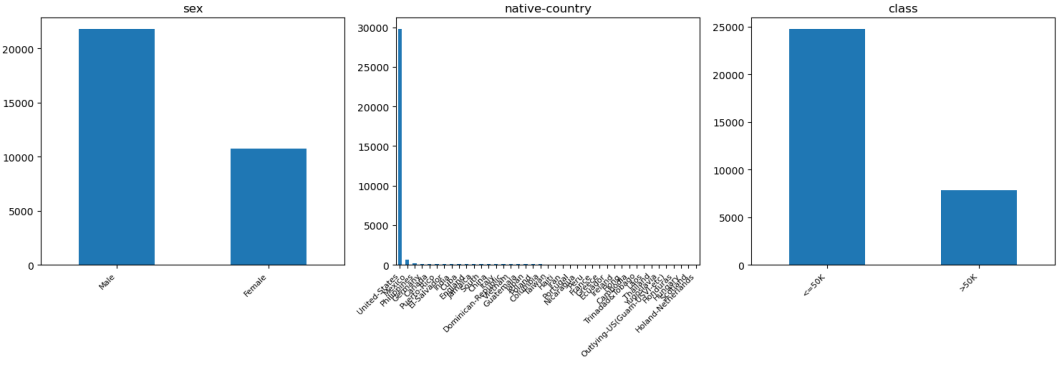
(a) 'age', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week'



(b) 'workclass', 'education', 'marital-status'



(c) 'occupation', 'relationship', 'race'



(d) 'sex', 'native-coutry', 'class'

Figure 3: (a) Distribution of numerical data. (b) (c) (d) Distribution of categorical data.

Looking at the data distribution for the categorical values we notice:

Column	Unique Values
workclass	8
education	16
marital-status	8
occupation	14
relationship	6
race	5
sex	2
native-country	41
class	2

Table 1: Unique values per categorical column.

There are around 40 unique values in the native country column, and more than 90% of the values were 'USA.' Therefore, by mapping the other values to 'Non-US,' we are able to reduce the number of unique values in the column, which would result in a high level of sparsity after one-hot encoding of the column. Moreover, when there are very few records of a certain value, such as 'Cambodia' in this case, it kind of makes those records outliers, which could negatively impact the performance of the model.
We split the data into train and test and we indicate which feature is the target.

```
# declaring the features and target
X = income_data.drop(['class'], axis=1)
y = income_data['class']
# splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↪  random_state=42)
```

Then we proceed to do a dummy encoding to all the categorical columns and we scale the data.

```
# dummy encoding the categorical columns
X_train = pd.get_dummies(X_train, drop_first=True)
X_test = pd.get_dummies(X_test, drop_first=True)
missing_cols = set(X_train.columns) - set(X_test.columns)
for col in missing_cols:
    X_test[col] = 0
X_test = X_test[X_train.columns]
# scaling the data
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Min-Max scaling rescales the values to a range between 0 and 1, which preserves the relationship between the values. Other normalization methods, such as z-score normalization or log transformation, may be appropriate for different types of data or analyses, but in this case, Min-Max scaling was sufficient.

2.2 Model Selection

We use a GridsearchCV to tune hyperparameters and optimize the learning models. Among the models used we have logistic regression, KNN, random forest, decision tree, gaussian naive bayes, gradient boosting, XGBoost, and others. The performance of each model was evaluated on the test dataset, and LightGBM was found to be the best model with approximately 87% accuracy.

On the test dataset, gradient boosting performed slightly better than lightGBM, as seen on Figure 4. However, in the cross-validation scores and the scores obtained after running lazy prediction, it was observed that lightGBM outperformed all other models in terms of performance. Moreover, lightGBM has some advantages over other boosting models, such as low model complexity, quick implementation, and better handling of large datasets. Therefore, considering all these factors, lightGBM was selected as the final model for this assignment.

	Model	Accuracy	Recall	Precision
6	Gradient Boosting	87.92	80.64	84.94
11	LightGBM	87.86	80.84	84.66
7	XGBoost	87.56	80.54	84.15
5	Random Forest	86.38	77.54	83.20
9	MLP Classifier	85.92	78.33	81.62
4	Decision Tree	85.72	76.11	82.49
0	Logistic Regression	85.58	77.17	81.46
2	SVC	85.55	76.67	81.67
10	SGD Classifier	85.31	76.79	81.02
1	KNN	83.46	74.67	77.97
3	Gaussian Naive Bayes	76.91	79.46	72.53
8	Perceptron	75.91	50.06	87.95

Figure 4: Models Comparison.

From this comparison, we can see the performance of boosting models has been better than the rest with models like random forest, MLP Classifier, etc not very much behind.

2.3 Customers and Revenue Prediction

First before applying the model, we need to clean the dataset again by dealing with the missing values (same as previously), hot encoding the categorical columns, scaling the data, and finally we can apply and deploy lightGBM after training.
After prediction, we get a total of 3234 customers with an income above 50k a year. And from that, we can use the formulas to calculate the final expected revenue for the company.

$$Revenue = totalRevenueHigh + totalRevenueLow - (costPerTargetedCustomer \times totalTargetedCustomers)$$

With

$costPerTargetedCustomer = 10$
$totalTargetedCustomers = 3234$
$totalRevenueHigh$ = Total revenue for all high class customers * acceptance rate * accuracy
$totalRevenueLow$ = Total cost for all low class customers * acceptance rate * (1-accuracy)

So when we use these formulas, we get a final total revenue of : $240251.27